Subproject R2: Phase 1, WP 4
Timing Predictability of Cache Replacement Policies

Reinhard Wilhelm

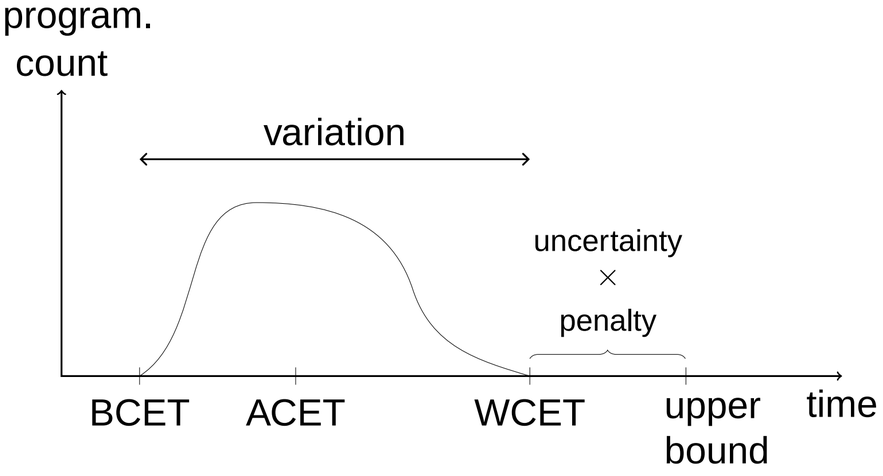
Introduction

Hard real-time systems must obey strict timing constraints. Therefore, one needs to derive guarantees on the worst-case execution times of a system's tasks. In this context, predictable behavior of system components is crucial for the derivation of tight and thus useful bounds.

We present results about the predictability of common cache replacement policies. To this end, we introduce three metrics, $evict$, $fill$, and mls that capture aspects of cache-state predictability. A thorough analysis of the LRU, FIFO, MRU, and PLRU policies yields the respective values under these metrics.

Timing Analysis

Goal: Determine upper bound on execution time of a program.



- Execution time depends on inputs and initial hardware state
- Modern processor features (caches, pipelining, speculation, etc.) increase variability of execution times of individual instructions;
  Timing accidents (cache miss) have large associated penalties (cache miss penalty)
- Static analyses exclude potential timing accidents, thereby reducing uncertainty
  $\Rightarrow$ Tighten upper bound on execution time

Cache Analysis

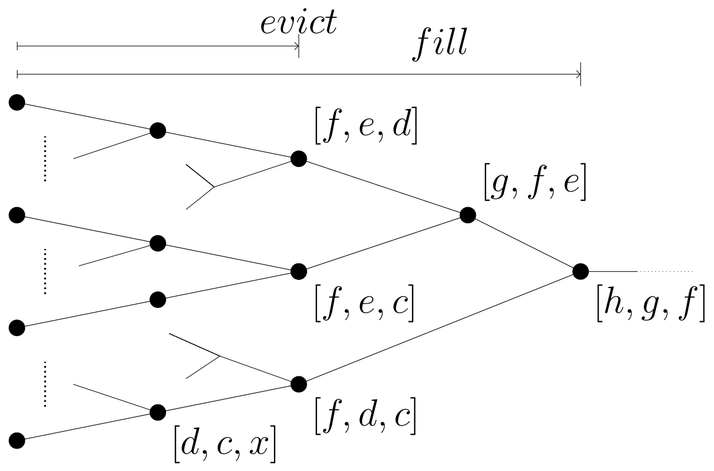
Goal: Predict whether a cache access is a miss or a hit. There are two concepts in static cache analysis: *may* and *must* information at program points.

- *Must* information is lower approximation of all possible cache contents: allows to predict hits (exclude timing accident cache miss)
- *May* information is upper approximation: allows to predict misses
- Intrinsic amount of uncertainty stemming for replacement policy

Metrics

We introduce *evict* and *fill*: Minimal numbers of cache accesses necessary (in all cases) to give definite answers to Hit/Miss-questions.

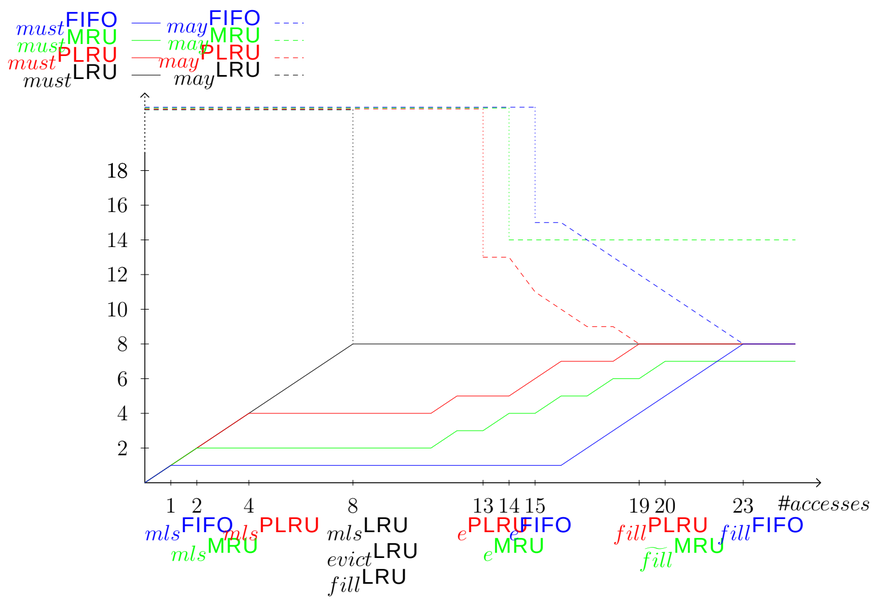
- Capture predictability of replacement policy
- Indicate how quickly knowledge about cache hits and misses can be (re-)obtained
- Mark limit on precision of *any* cache analysis



Third metric *minimal life-span (mls)* tells how much must-infomation can be obtained easily, i.e. by a cheap must-analysis.

Results

The figure shows the milestones $evict$ and $fill$ as well as the evolution of may and must information with the number of memory accesses increasing.



- LRU is the most predictable policy, no other policy can do better
- PLRU is significantly worse to predict
- MRU is special since full information may never be reached
- FIFO is worst with very low must-information for a large number of accesses

Reference: Timing Predictability of Cache Replacement Policies, Real-Time Systems Journal 2007, to appear

SFB / TR 14 AVACS
Automatic Verification and Analysis of Complex Systems
Carl von Ossietzky Universität Oldenburg
Albert-Ludwigs Universität Freiburg
Universität des Saarlandes, Saarbrücken
Max-Planck-Institut für Informatik, Saarbrücken